▨ 연구논문

후보 2-항목집합의 개수를 최소화한 연관규칙 탐사 알고리즘

An Algorithm for Mining Association Rules by Minimizing the Number of Candidate 2-Itemset

황 종 원*
Hwang, Jong Won
강 맹 규**
Kang, Maing Kyu

Abstract

Mining for association rules between items in a large database of sales transaction has been described as an important data mining problem. The mining of association rules can be mapped into the problem of discovering large itemsets. In this paper we present an efficient algorithm for mining association rules by minimizing the total numbers of candidate 2-itemset, $|C_2|$. More the total numbers of candidate 2-itemset, less the time of executing the algorithm for mining association rules. The total performance of algorithm depends on the time of finding large 2-itemsets. Hence, minimizing the total numbers of candidate 2-itemset is very important. We have performed extensive experiments and compared the performance of our algorithm with the DHP algorithm, the best existing algorithm.

1. 서론

정보시스템들이 보유하고 있는 데이터베이스의 규모도 기가 바이트 단위에서 테라 바이트 단위 시대로 접어들게 되었다. 이러한 대용량 데이터베이스에서 의사결정에 필요한 지식을 발견하고자 하는 연구가 진행 중이다.

본 연구에서는 대용량 데이터베이스에서의 지식 발견(KDD : knowledge discovery in databases)과정 중 데이터 마이닝 단계에서 필요한 탐사 알고리즘을 제안한다. 데이터 마이닝

* 한양대학교 산업공학과 박사과정
** 한양대학교 산업공학과

(data mining)은 지식을 발견하고자 하고자 하는 목표 데이터(target data)로부터 흥미 있는 패턴을 발견하기 위해서 발견 기법을 적용시켜 패턴을 추론하는 KDD과정의 한 단계이다[3, 10]. 데이터 마이닝에는 연관규칙 탐사(association rule mining), 순차패턴 탐사(sequential pattern mining), 분류(classification), 군집 구분(clustering), 데이터 요약(summarization) 등이 있다[7].

본 연구에서는 연관규칙을 탐사하는 효율적인 알고리즘을 제안한다. Agrawal 등[5]은 연관규칙을 탐사하는 문제를 처음으로 소개하였다. 연관규칙 탐사는 동시에 발생하는 사건 그룹 내에서 사건들 사이에 존재하는 친화성 또는 패턴을 찾아낸다. 연관규칙 탐사의 대표적인 예로 IBM의 '시장 바구니 분석(market basket analysis)을 들 수 있다. 이것은 슈퍼마켓에서 소비자들이 구입한 물품들의 목록을 분석함으로써 분유를 구매하는 경우의 10%는 소비자들이 맥주도 함께 구매한다는 것과 같은 패턴을 발견하는 것이다.

연관규칙를 탐사하는 알고리즘은 상품 진열, 통신 네트워크 고장의 조기 경보, 데이터베이스 마케팅, 인터넷 환경에서 사용자의 항해 경로를 분석하는 업무 등에 적용되고 있다[8, 12].

연관규칙 탐사는 처음 두 번의 반복 시행에서 빈발 2-항목집합을 탐사하는 비용이 모든 빈발 항목집합을 탐사하는 총 비용을 좌우하는 것으로 알려져 있다[14], 본 연구에서는 후보 2-항목집합의 총 개수를 최소화하여 메모리 상의 연산 시간을 줄이는 효율적인 알고리즘을 제안한다.

2. 연관규칙 탐사 알고리즘

2.1 문제 정의

$I = \{i_1, i_2, ..., i_m\}$는 항목(item)이라 불리는 문자들의 집합이다. D는 트랜잭션들의 집합이고 각 트랜잭션 T는 $T \subseteq I$인 항목들의 집합이다. 각 트랜잭션에서 발생하는 항목들의 개수는 고려하지 않는다. 각 트랜잭션은 식별자(transaction identifier : TID)를 가지고 있다. X는 항목들의 집합 즉, 항목집합(itemset)이다. 트랜잭션 T가 X를 포함한다고 하면, $X \subseteq T$이다. 연관규칙은 $X \rightarrow Y$의 형식으로 나타내고, 여기서, $X \subset I$, $Y \subset I$, 및 $X \cap Y = \varnothing$ 이다.

"트랜잭션 집합인 D에서 $X \rightarrow Y$는 지지도(support) s를 갖는다"의 의미는 D의 트랜잭션들 중 s%는 $X \cup Y$를 포함하고 있음을 의미한다. 또한, 데이터베이스 D중 s%의 트랜잭션들이 항목집합 $X \cup Y$를 지지한다고 한다. "트랜잭션 집합 D에서 $X \rightarrow Y$는 신뢰도(confidence) c를 가지고 있다"의 의미는 X를 포함하는 D의 트랜잭션들 중 c%가 Y 또한 포함하고 있음을 의미한다.

연관규칙을 탐사하는 문제는 연관규칙의 지지도와 신뢰도가 최소 지지도(minimum support)와 최소 신뢰도(minimum confidence)보다 크거나 같은 모든 연관규칙을 찾는 것이다. 최소지지도와 최소 신뢰도는 사용자가 정의한 값이다. 전체 데이터베이스 D에 대해서 D의 트랜잭션 총 개수*최소지지도를 최소지지도 개수라고 한다. D의 부분 데이터베이스 D^1에 대해 D^1의 트랜잭션 총 개수*최소지지도를 부분 최소지지도 개수라고 한다.

최소지지도(또는 최소지지도 개수) 이상을 갖는 항목집합을 빈발 항목집합(large itemset 또는 frequent itemset)이라 한다. k개의 항목들로 이루어진 빈발 항목집합을 빈발 k-항목집합이라 한다. 빈발 k-항목집합들의 집합을 L_k라 하고 이를 위한 후보 k-항목집합들을 C_k라 한다. 후보 k-항목집합은 데이터베이스를 액세스하여 지지도 개수를 계산하기 직전의 k-항목집합이다.

연관규칙을 발견하는 과정은 다음의 두 단계로 구성된다[7]. 첫 번째 단계에서는 빈발 항목집합을 탐사한다. 주어진 최소지지도 이상의 트랜잭션들이 지지하는 모든 항목집합들을 탐사한다. 두 번째 단계에서는 빈발 항목집합들을 이용하여 연관규칙을 찾는다.

연관규칙을 탐사하는 두 단계 중에서 전체 수행속도에 가장 큰 영향을 미치는 것은 첫 번째 단계이다. 따라서 기존의 연관 규칙 탐사 알고리즘은 빈발 항목집합을 찾는 첫 번째 단계에 초점을 두고 있다. 본 연구는 첫 번째 단계를 위한 효율적인 알고리즘을 제안한다.

2.2 기호 정의

본 연구에서 사용되는 기호는 다음과 같다.

D : 트랜잭션 T로 구성된 전체 데이터베이스
D^1 : 부분 빈발 1-항목집합을 발견하기 위한 트랜잭션 T로 구성된 부분 데이터베이스($D^1 \subset D$)
D^2 : 전체 데이터베이스 D중 D_1을 제외한 나머지 트랜잭션 T로 구성된 부분 데이터베이스($D^2 \subset D$, $D^1 \cap D^2 = \varnothing$)
C_k : 모든 후보 k-항목집합들의 집합
$|C_k|$: C_k의 원소인 후보 k-항목집합들의 총 개수
L_k : 모든 빈발 k-항목집합들의 집합
$|L_k|$: L_k의 원소인 빈발 k-항목집합들의 총 개수
L_{d1} : 부분 빈발 1-항목집합들의 집합
H_2 : L_1을 발견할 때 생성한 해시 테이블
$h_1(x)$: 항목집합 x에 적용하는 첫 번째 해시 함수
$h_2(x)$: 항목집합 x에 적용하는 두 번째 해시 함수
H_T : 해시 트리의 해시 테이블
$h_T(i)$: H_T에서 항목 I에 적용하는 해시 함수
c : 항목
L_{m1} : L_1의 원소이고 L_{d1}의 원소가 아닌 빈발 1-항목집합들의 집합

2.3 기존 연구

Apriori 알고리즘[4]은 후보 생성을 위한 결합 단계와 지지도 계산 단계로 구성된다. 지지도 계산 단계에서는 데이터베이스를 액세스하여 각 후보 항목집합의 지지도 개수를 계산하여 최소지지도 개수 이상을 갖는 빈발 항목집합을 생성한다. 데이터베이스를 액세스함으로써 다음 결합 단계에서 생성할 후보 항목집합의 개수를 감소시킨다.

DHP(direct hashing and pruning)[14] 알고리즘은 Apriori알고리즘과 같이 L_{k-1}로부터 C_k 를 생성한다. 그러나 DHP 알고리즘이 사용한 해시 기법은 후보 항목집합의 수를 최소화한다. 특히 L_2를 위한 C_2 의 크기가 기존 방법에 비해 매우 적다. 그렇게 함으로써 전체 프로세스의 수행상의 병목을 개선한다. 연관규칙의 특성을 이용함으로써 트랜잭션의 수뿐만 아니라 각 트랜잭션 항목의 수 또한 감소시킬 수 있음을 보여 주었다.

윤유경 등[2]은 시간 흐름에 따른 연관규칙과 연관규칙 사이의 관계를 발견하는 방법을 제시하였다. 생성된 모든 빈발 항목집합의 시간 흐름에 따른 변화를 알 수 있다. 시간의 관점과 빈발 항목집합의 연합 단계의 관점에서 빈발 항목집합의 변화를 비교한다. 시간 지지도를 정의하여 빈발 항목집합 간의 비교를 수치화하였다.

그 밖에 연관규칙의 탐사에서는 정량적 데이터[16], 일반화[15], 다중수준(multiple level)[11], 병렬 처리(parallel processing)[6], 분산 환경, 점진적 갱신(incremental updating)[1], 데이터 큐브(data cube)[13] 등 다양한 환경을 고려한 여러 알고리즘들이 제시되었다.

3. 제안하는 알고리즘

3.1 기본 개념과 주요 절차

후보 항목집합의 개수가 많으면 많을수록 빈발 항목집합을 탐사하는데 소요되는 연산 비용은 더욱 증가한다[14]. 특히, 처음 두 번의 반복 시행에서의 연산 비용(L_1과 L_2의 발견 비용)은 모든 빈발 항목집합을 발견하는 총 비용을 좌우한다. 따라서, 본 연구에서는 처음 두 번의 반복 시행에서 후보 2-항목집합의 총 개수 $|C_2|$을 최소화하고 지지도 개수를 효율적으로 계산하는 알고리즘을 제안한다.

그림 3.1은 제안하는 알고리즘의 주요 절차를 나타낸다.

```
모든 항목 c의 지지도 개수를 0으로 초기화한다.
해시 테이블 H2의 모든 레코드을 0으로 초기화한다.
데이터베이스 D1과 D2 를 액세스한 트랜잭션 개수의 누적값 i 를 0으로 초기화한다.
모든 (트랜잭션 T ∈ D1) {
   모든 (항목 c ∈ T) { 항목 c의 지지도 개수를 1 증가시킨다. }
   모든 (2-항목집합 x ⊂  T) {
      만약 (i < 트랜잭션의 총 개수/2)
         H2[h1(x)]  = H2[h1(x)] + 1
      그렇지 않으면
         H2[h2(x)]  = H2[h2(x)] + 1                .
   }
   i = i + 1
}
L dl = {항목 c | 항목 c의 지지도 개수 >= D1에 대한 부분 최소지지도 개수}
모든 (트랜잭션T ∈ D2 ) {
   모든 (항목 c ∈ T) { 항목 c의 지지도 개수를 1 증가시킨다. }
   트랜잭션 T는 L dl에 속하는 항목들로 구성된다.
   모든 (2-항목집합 x ⊂ T) {
      만약 (i < 트랜잭션의 총 개수/2)
         H2[h1(x)] = H2[h1(x)] + 1.
      그렇지 않으면
         H2[h2(x)] = H2[h2(x)] + 1.
   }
   i = i + 1
}
L1 = {항목 c | 항목 c의 지지도 개수 >= D에 대한 최소지지도 개수}
모든 (Lml*L 1로 생성한 2-항목집합 x ⊂ T) {
   H2[h1(x)] = 최소지지도 개수/2
   H2[h2(x)] = 최소지지도 개수/2
}
k = 2
후보 2-항목집합을 생성한다.
만약 (| Ck |)  {
   만약 (k == 2)
      후보 2-항목집합의 지지도 개수를 계산한다.
   그렇지 않으면
      후보 k-항목집합의 지지도 개수를 계산하고 데이터베이스를 축소한다
         (알고리즘 DHP와 동일).
   Lk = {후보 k-항목집합 | 후보 k-항목집합의 지지도 개수 >= 최소지지도 개수}
   k = k + 1;
   후보 k-항목집합을 생성한다(알고리즘 DHP와 동일).
}
```

```
후보 2-항목집합을 생성한다.
{
   모든 (L1*L1로 생성한 2-항목집합 x)
      만약 (H2[h1(x)] + H2[h2(x)] >= 최소지지도 개수)
         C2 = C2 ∪ {x}
}

후보 2-항목집합의 지지도 개수를 계산한다
{
   모든 (트랜잭션 T ∈ D) {  // 데이터베이스 D 를 액세스한다
      메모리 상의 트랜잭션 T는 L 1에 속하는 항목들로 구성된다.
      모든 (2-항목집합 x ⊂ T)
         만약 (H2[h1(x)] + H2[h2(x)] >= 최소지지도 개수)
            x의 지지도 개수를 1 증가시킨다.
   데이터베이스 D를 축소한다(알고리즘 DHP와 동일).
}
```

그림 3.1 제안하는 알고리즘의 주요 절차

3.2 메모리 상의 트랜잭션의 크기를 축소하는 방법

3.2.1 기본개념

기존의 DHP 알고리즘은 L_1을 발견하기 위해서 데이터베이스를 액세스 할 때, 각 트랜잭션의 모든 항목들을 일단 메모리에 저장한다. 그리고 나서 각 트랜잭션으로부터 생성시킬 수 있는 모든 2-항목집합에 대해서 해시 함수를 적용한다. L_1에 참여하지 않는 1-항목을 포함한 2-항목집합에 대해서도 해시 함수를 적용한다. 빈발 2-항목집합이 아니면서 해시 테이블 H_2의 레코드 값이 최소지지도 개수보다 큰 값을 갖는 후보 2-항목집합들이 많이 존재할 수 있다. 따라서 후보 2-항목집합의 총 개수 $|C_2|$이 증가한다.

본 연구에서는 메모리 상의 트랜잭션의 크기를 축소하여 트랜잭션의 2-항목집합에 해시 함수를 불필요하게 적용하지 않는 방법을 제안한다. 그렇게 함으로써 $|C_2|$을 감소시킨다. 따라서, 데이터베이스도 DHP 알고리즘보다 더 축소할 수 있고 지지도 개수를 계산하는 시간을 단축할 수 있다.

사용자가 정의한 부분 데이터베이스 D^1의 트랜잭션의 개수만큼을 액세스할 때는 DHP 알고리즘과 동일한 방식으로 C_1의 지지도 개수를 계산하고 해시 함수를 적용한다. 지지도 개수가 D^1에 대한 부분 최소지지도 개수 보다 크거나 같은 후보 1-항목집합들의 집합, 즉 부분 빈발 1-항목집합들의 집합 L_{d1}을 발견한다. 전체 데이터베이스 D 중 D^1을 제외한 부분 데이터베이스 D^2의 트랜잭션를 액세스할 경우는 L_{d1}에 속하지 않는 항목은 메모리에 저장하지 않는다. 메모리 상에 축소된 트랜잭션의 2-항목집합에 대해서만 해시 함수를 적용한다.

L_{d1}과 L_1으로부터 2-항목집합을 생성한다. 이렇게 생성한 2-항목집합의 해시 테이블의 레코드 값에 최소지지도 개수를 할당한다.

데이터베이스를 두 번째 액세스할 때, 각 트랜잭션에 대해서 L_1에 속하지 않는 항목을 제외한 나머지 항목만 메모리에 저장한다. 메모리 상에 축소된 트랜잭션에 대해서 후보 2-항목집합의 지지도 개수를 계산한다.

3.2.2 예

그림 3.2는 전체 데이터베이스이다. 최소지지도 개수는 2이다. 부분 데이터베이스 D^1의 트랜잭션의 총 개수는 3이고 부분 최소지지도 개수는 1이다. 해시 함수를 (첫 번째 항목 + 두 번째

항목)*7/13인 연산을 한 후에 연산의 정수형 나머지 값을 반환하는 함수로 정의한다. 해시 함수 값을 반환되는 정수형 나머지 값이다. 데이터베이스를 처음 액세스할 때, D^1의 트랜잭션(트랜잭션 식별자가 100, 200, 300인 트랜잭션)만 액세스하여 일차적으로 부분 빈발 1-항목집합들의 집합 L_{d1}을 생성한다. 그림 3.3은 D^1의 트랜잭션만 액세스한 경우의 후보 1-항목집합과 그 지지도 개수를 나타낸다. 부분 최소지지도 개수 1보다 크거나 같은 지지도 개수를 갖는 후보 1-항목집합 {1}, {3}, {4}, {5}, {6}으로 구성된 L_{d1}을 생성한다.

트랜잭션 식별자	항목
100	3, 4
200	1, 5, 6
300	1, 5, 6
400	1, 5, 6
500	0, 1, 2, 3, 4
600	4, 5, 6, 7

그림 3.2 전체 데이터베이스

후보 1-항목집합	지지도 개수
{0}	0
{1}	2
{2}	0
{3}	1
{4}	1
{5}	2
{6}	2
{7}	0

그림 3.3 부분 데이터베이스 D^1을 액세스한 후의 후보 1-항목집합의 지지도 개수

그림 3.4는 기존의 DHP 알고리즘에서, 그림 3.5는 메모리 상의 트랜잭션을 축소하는 방법에서 각 트랜잭션의 메모리 상에 저장될 항목과 해시 함수를 적용할 2-항목집합을 각각 나타낸다. 그림 3.5의 트랜잭션 500을 예를 들어 설명하면 메모리 상에 항목1, 3, 4만을 저장한다. L_{d1}에 속하지 않는 항목 0, 2는 제외된다. 또한, 항목 1, 3, 4에 대해서만 해시 함수를 적용한다.

트랜잭션식별자	메모리상에 저장된 항목	해시 함수를 적용할 2-항목집합
100	3, 4	{3, 4}
200	1, 5, 6	{1, 5} {1, 6} {5, 6}
300	1, 5, 6	{1, 5} {1, 6} {5, 6}
400	1, 5, 6	{1, 5} {1, 6} {5, 6}
500	0, 1, 2, 3, 4	{0, 1} {0, 2} {0, 3} {0, 4} {1, 2} {1, 3} {1, 4} {2, 3} {2, 4} {3, 4}
600	4, 5, 6, 7	{4, 5} [4, 6} {4, 7} {5, 6} {5, 7} {6, 7}

그림 3.4 DHP 알고리즘의 메모리 상의 트랜잭션과 해시 함수를 적용할 2-항목집합

트랜잭션식별자	메모리상에 저장된 항목	해시 함수를 적용할 2-항목집합
100	3, 4	{3, 4}
200	1, 5, 6	{1, 5} {1, 6} {5, 6}
300	1, 5, 6	{1, 5} {1, 6} {5, 6}
400	1, 5, 6	{1, 5} {1, 6} {5, 6}
500	1, 3, 4	{1, 3} {1, 4} {3, 4}
600	4, 5, 6	{4, 5} [4, 6} {5, 6}

그림 3.5 제안하는 알고리즘의 메모리 상의 트랜잭션과 해시 함수를 적용할 2-항목집합

그림 3.6은 DHP 알고리즘과 메모리 상의 트랜잭션을 축소하는 방법에서 해시 함수를 적용시키지 않은 후보 2-항목집합의 해시 함수 값과 해시 테이블의 레코드 값을 나타낸다. 그림 3.7은 DHP 알고리즘에서 해시 함수를 통하여 생성한 2-항목집합이다. 반면, 그림 3.8은 메모리 상의 트랜잭션을 축소하는 방법에서 해시 함수를 적용하여 생성한 후보 2-항목집합이다. DHP

알고리즘에서는 후보 2-항목집합의 개수가 6개인 반면 메모리 상의 트랜잭션을 축소하는 방법에서는 4개이다.

2-항목 집합	해시 함수 값	DHP알고리즘의 해시 테이블의 레코드 값	제안하는 알고리즘의 해시 테이블의 레코드 값
{1, 3}	2	2	1
{1, 4}	9	2	1
{1, 5}	3	4	3
{1, 6}	10	5	5
{3, 4}	10	5	5
{3, 5}	4	0	0
{3, 6}	11	1	1
{4, 5}	11	1	1
{4, 6}	5	1	1
{5, 6}	12	5	4

그림 3.6 2-항목집합의 해시 함수 값과 해시 테이블의 레코드 값

후보 2-항목집합	지지도 개수
{1, 3]	1
{1, 4}	1
{1, 5}	3
{1, 6}	3
{3, 4}	2
{5, 6}	4

그림 3.7 DHP 알고리즘의 후보 2-항목집합

후보 2-항목집합	지지도 개수
{1, 5}	3
{1, 6}	3
{3, 4}	2
{5, 6}	4

그림 3.8 제안하는 알고리즘의 후보 2-항목집합

3.3 해시 테이블을 참조한 후 지지도 개수를 계산하는 방법

3.3.1 기본 개념

본 연구에서는 메모리상의 트랜잭션이 축소된 사실과 충돌을 일으키는 동의어인 후보 2-항목집합의 수가 감소된 사실을 이용하여 효율적으로 지지도 개수를 계산하는 방법을 제안한다.

데이터베이스를 액세스하여 후보 2-항목집합의 지지도 개수를 계산할 때, 메모리 상의 축소된 트랜잭션에 대하여 모든 2-항목집합을 생성한다. 생성한 모든 2-항목집합 x에 대해서 해시 함수 h(x)를 적용하여 해시 테이블 H_2 레코드 값이 최소지지도 개수보다 크거나 같은지를 검사한다. H_2의 레코드 값이 최소지지도 개수보다 크거나 같은 2-항목집합만 메모리에 저장한다. 메모리에 저장한 2-항목집합의 각 항목 i에 대해서 해시 트리의 해시 테이블 H_T에 대한 해시 함수 $h_T(i)$를 적용하여 지지도 개수를 계산한다. 그렇게 함으로써 지지도 개수 계산을 위한 탐색 공간을 축소한다.

3.3.2 예

3.2.2 예에서 사용한 전체 데이터베이스, 최소지지도 개수, 해시 함수를 사용한다. 해시 트리의 해시 테이블 H_T의 크기는 5이고 해시 함수는 "mod 5"이다.

DHP 알고리즘에서는 트랜잭션 500 {0, 1, 2, 3, 4}의 경우, 트랜잭션의 각 항목에 대해서 해시 함수 "mod 5"를 적용한다. 해시 테이블 H_T의 레코드 0, 1, 2, 3, 4가 가리키는 후보 2-항목집합의 지지도 개수를 계산한다. 따라서 트랜잭션 500에 대해서는 후보 2-항목집합 {5, 6}, {1, 5}, {1, 6}, {3, 4}의 지지도 개수를 계산한다.

해시 테이블 H_2를 참조하여 지지도를 계산하는 방법에서는 메모리 상에 축소된 트랜잭션 500 {1, 3, 4}에 대해서 먼저 2-항목집합을 생성한다. 2-항목집합 {1, 3}, {1, 4}, {3, 4}에 대해서 해시 함수를 적용하여 해시 테이블 H_2의 레코드 값이 최소지지도 개수 2보다 크거나 같은지를 검사한다. 그림 3.6에서 보여주듯이 {3, 4}의 해시 테이블 레코드 값 5만이 최소지지도 개수보다 크거나 같다. 따라서, 첫 항목 3에 대해서 해시 함수 "mod 5"를 적용하여 해시 테이블 H_T의 레코드 3이 가리키는 후보 2-항목집합의 지지도 개수를 계산한다. 따라서 트랜잭션 500에 대해서는 후보 2-항목집합 {3, 4}의 지지도 개수를 계산한다.

3.4 두 개의 해시 함수를 적용하는 방법

3.4.1 기본개념

DHP 알고리즘은 후보 2-항목집합의 총 개수 $|C_2|$를 줄이기 위해서 하나의 해시 함수를 적용한다. 특정 2-항목집합이 빈발 2-항목집합인 경우 그 특정 2-항목집합과 같은 주소로 할당되어 충돌이 발생하는 빈발 2-항목집합이 아닌 2-항목집합이 많이 존재할 수 있다. 이러한 경우 빈발 2-항목집합이 아닌 2-항목집합들은 해시 테이블 H_2의 레코드 값이 최소지지도 개수보다 크거나 같게 된다. 따라서 $|C_2|$은 증가한다.

본 연구에서는 데이터베이스의 전체 트랜잭션 중 처음 1/2와 다음 1/2에 각각 다른 해시 함수를 적용한다. L_1을 결합하여 C_2를 생성할 때 첫 번째 해시 함수의 레코드 값과 두 번째 해시 함수의 레코드 값을 더한 값이 최소지지도 개수보다 작으면 후보 2-항목집합이 될 수 없다. 그렇게 함으로써 빈발 2-항목집합과 충돌이 발생하면 무조건 후보 2-항목집합에 등록되는 단점을 보안한다. 알고리즘 DHP에 비해서 후보 2-항목집합의 총 개수 $|C_2|$를 최소화하다.

3.4.2 예

그림 3.9는 전체 데이터베이스이다. 최소지지도 개수는 4이다. 첫 번째 해시 함수는 (첫 번째 항목*두 번째 항목)/13인 연산을 한 후에 연산의 정수형 나머지 값을 반환하는 함수이다. 두 번째 해시 함수는 (첫 번째 항목 + 두 번째 항목)/13인 연산을 한 후에 연산의 정수형 나머지 값을 반환하는 함수이다. 해시 테이블 H_2을 생성할 때 트랜잭션 100과 200은 첫 번째 해시 함수를 사용하고 트랜잭션 300과 400은 두 번째 해시 함수를 사용한다.

트랜잭션 식별자	항목
100	1, 6, 8
200	2, 3
300	4, 8
400	1, 19

그림 3.9 전체 데이터베이스

그림 3.10은 기존의 DHP알고리즘과 마찬가지로 첫 번째 해시 함수 하나만 적용한 경우의 해시 테이블이다. 그림 3.11은 서로 다른 두 개의 해시 함수를 적용한 경우의 해시 테이블이다. DHP 알고리즘의 경우 해시 테이블 H_2에서 최소지지도 개수보다 크거나 같은 레코드 값을 갖는 2-항목집합의 개수는 4개이다. 반면, 제안하는 알고리즘의 경우 2-항목집합은 하나도 없다.

						{1, 19} {4, 8} {2, 3} {1, 6}		{1, 8}	{6, 8}			
0	1	2	3	4	5	6	7	8	9	10	11	12

그림 3.10 DHP 알고리즘의 해시 테이블

						{2, 3} {1, 6}	{1, 19}	{1, 8}	{6, 8}			{4, 8}
0	1	2	3	4	5	6	7	8	9	10	11	12

그림 3.11 제안하는 알고리즘의 해시 테이블

4. 실험

본 장에서는 실험 결과를 통하여 제안하는 알고리즘과 기존의 DHP 알고리즘의 실행 시간을 비교한다. Visual C^{++} 5.0을 사용하여 알고리즘을 구현한다. 실험은 CPU 200MHz, 메인 메모리 64M-byte를 가진 데스크탑 PC상에서 수행한다.

4.1 실험 데이터의 생성

본 연구에서 실험 데이터를 생성하기 위해서 사용한 방법은 Agrawal 등[4]이 제시한 방법과 같다. 데이터베이스 T10.I4.D100은 각각 트랜잭션에 있는 항목들의 평균 개수가 10이고, 최대 잠재적인 빈발 항목집합들의 평균 크기 4이고, 데이터베이스의 트랜잭션의 총 개수가 100,000임을 의미한다. 항목의 총 개수 N = 1000, 최대 잠재적인 빈발 항목집합들의 개수 |L| = 10000으로 설정한다. 실험에 사용되는 해시 테이블 H_2의 크기는 DHP 알고리즘과 마찬가지로 2^{19} 이다.

4.2 트랜잭션 총 개수의 증가에 따른 실행 시간 비교

데이터베이스 T15I4에 대해서 최소지지도를 0.3%로 설정하고 트랜잭션의 총 개수를 100,000, 200,000, 400,000, 600,000, 800,000, 1,000,000까지 증가시키면서 트랜잭션의 총 개수의 증가에 따른 실행 시간을 평가한다. 최종적으로 빈발 6-항목집합까지 생성한다.

트랜잭션의 총 개수(천)	100	200	400	600	800	1000
DHP(초)	86	172	356	534	709	919
제안하는 해법(초)	70	139	292	435	581	743

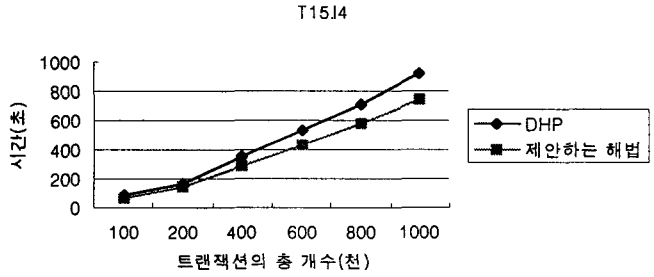

그림 4.1 트랜잭션의 총 개수 증가에 따른 실행 시간 비교

그림 4.1은 트랜잭션 총 개수의 증가에 대해서 제안하는 해법의 실행시간이 DHP 알고리즘보다 완만하게 증가함을 보여준다.

4.3 지지도 감소에 따른 실행 시간 비교

데이터베이스 T10I4D1000에 대해서 최소지지도를 0.2, 0.15, 0.125, 0.1, 0.075%까지 감소시키면서 지지도의 감소에 따른 실행 시간을 평가한다.

지지도(천)	0.2	0.15	0.125	0.1	0.075
DHP(초)	333	505	665	1103	1854
제안하는 해법(초)	277	431	597	1046	1825
최종 발견한 L_k의 k	6	7	9	9	10

그림 4.2 데이터베이스 T10I4D1000에 대해서
지지도의 감소에 따른 실행 시간 비교

데이터베이스 T15I4D1000에 대해서 최소지지도를 0.4, 0.35, 0.3, 0.25, 0.2%까지 감소시키면서 지지도의 감소에 따른 실행 시간을 평가한다..

지지도(천)	0.4	0.35	0.3	0.25	0.2
DHP(초)	565	696	888	1330	2258
제안하는 해법(초)	454	530	749	1154	2177
최종 발견한 L_k의 k	3	3	6	6	7

그림 4.3 데이터베이스 T15I4D1000에 대해서
지지도의 감소에 따른 실행 시간 비교

그림 4.2와 그림 4.3은 지지도 감소에 대해서 제안하는 해법의 실행 시간이 DHP 알고리즘보다 빠른 것을 보여준다.

4.4 후보 2-항목집합의 총 개수의 변화

데이터베이스 T15I4D1000에 대해서 최소지지도를 0.4, 0.35, 0.3, 0.25, 0.2%까지 감소시키면서 지지도의 감소에 따른 후보 2-항목집합의 총 개수 $|C_2|$의 변화를 비교한다,

지지도(천)	0.4	0.35	0.3	0.25	0.2
DHP의 $\|C_2\|$	20,459	26,682	34,536	43,926	57,394
제안하는 해법의 $\|C_2\|$	6,100	10,623	17,788	28,674	45,565

그림 4.4 데이터베이스 T15I4D1000에 대해서
지지도의 감소에 후보 2-항목집합의 총 개수 $|C_2|$의 변화

그림 4.4는 제안하는 해법이 DHP 알고리즘에 비해 $|C_2|$를 매우 감소시킨 것을 보여준다. 특히 지지도가 0.4인 경우는 $|C_2|$를 70%까지 감소시킨다.

이상의 실험결과와 같이 본 연구에서 제안하는 알고리즘은 $|C_2|$를 최소화함으로써 기존의 DHP 알고리즘보다 좋은 성능(performance)를 보여준다.

5. 결론

대용량 데이터베이스에서 지식을 발견하는 KDD 과정 중 데이터 마이닝 단계에서의 연관규칙 탐사에 관한 연구가 활발하다. 본 연구에서는 후보 2-항목집합의 총 개수를 최소화함으로써 성능을 향상시킨 효율적인 연관규칙 탐사 알고리즘을 제안한다.

본 연구에서 제안하는 알고리즘은 다음과 같은 세 가지 방법들로 구성되어 있다. 첫째, 메모리 상의 트랜잭션의 크기를 축소하여 해시 함수를 호출하는 비용과 지지도 개수를 계산하는 비용을 감소한다. 둘째, 후보 2-항목집합의 지지도 개수를 계산할 때, 해시 테이블을 참조함으로써 지지도를 계산하는 후보 2-항목집합의 탐색 범위를 축소한다. 셋째, 두 개의 해시 함수를 적용하여 후보 2-항목집합의 총 개수를 최소화한다.

다양한 실험을 통하여 기존의 DHP알고리즘과 제안하는 알고리즘을 비교하였다. 제안하는 알고리즘은 DHP 알고리즘에 비해 후보 2-항목집합의 총 개수를 많게는 70%까지 감소시킨다. 그 결과 제안하는 알고리즘은 기존의 연관규칙 탐사 알고리즘보다 좋은 성능을 보여준다.

참고 문헌

1. 김동필, 지영근, 황종원, 강맹규, "데이터 마이닝에서 기존의 연관규칙을 갱신하는 효율적인 알고리즘 개발," 공업경영학회, 제21권, 제45집, 1998.
2. 윤유경, 윤종필, "대용량 데이터베이스에서 시간 흐름에 따른 연관 항목 집합의 변화 탐색," 데이터베이스연구회지, 제12권, 제4호, pp. 23-42, 1996.
3. Adriaans, P. and D. Zantinge, *Data Mining,* Addison-Wesley, England, 1996.
4. Agrawal, R. and R. Srikant, "Fast Algorithms for Mining Association Rules," *Proc. the 20th VLDB Conference,* 1994.
5. Agrawal, R., T. Imielinski, and A. Swami, "Mining Association Rules between Sets of Items in Large Databases," *Proc. the 1993 ACM SIGMOD Conference,* pp. 207-216, May 1993.
6. Agrawal, R. and J. C. Shafer, "Parallel Mining of Association Rules: Design, Implemention and Experience," IBM Research Report RJ 10004, January 1996.
7. Chen, M. S., J. Han, and P. S. Yu, "Data Mining: An Overview from Database Perspective," *IEEE Transaction on Knowledge and Data Engineering*, Vol.8, No.6, December 1996.
8. Chen, M. S., J. S. Park, and P. S. Yu, "Data Mining for Path Traversal Patterns in a Web Environment," *Proc. the 16th International Conference on Distributed Computing Systems*, May 1996.
9. Cheung, D. W., J. Han, V. T. Ng, and C. Y. Wong, "Maintenance of Discovered Rules in Large Databases: An Incremental Updating Technique," *Proc. 12th International Conference on Data Engineering*, pp. 106-114, February 1996.
10. Fayyad U. M., G. Piatestsky-Shapiro, and P. Smyth, "From Data Mining to Knowledge Discovery: An Overview," In *Adavancences in Knowledge Discovery and Data Mining*, Fayyad U. M., et al, eds., AAAI/MIT, Menlo Park, CA, pp. 1-34, 1996.
11. Fu, Y., "Discovery of Multiple-Level Rules from Large Databases," PhD dissertation, Simon Fraser University, Canada, July 1996.
12. Hatonen, K., M. Klemettinen, H. Mannila, P. Ronkainen, and H. Toivonen, "Knowledge Discovery from Telecommunication Network Alarm Databases," *Proc. 12th International Conference on Data Engineering*, pp. 115-122, February 1996.
13. Kamber M., J. Han, and J. Y. Chiang, "Using Data Cubes for Metarule-Guided Mining of Multi-Dimensional Association Rules," Technical Report CS-TR 97-10, School of Computing Science, Simon Fraser University, Canada, May 1997.
14. Park, J. S., M. S. Chen, and P. S. Yu, "An Effective Hash-Based Algorithm for Mining Association Rules," *Proc. ACM SIGMOD International Conference on Management of Data*, pp. 175-186, May 1995.
15. Srikant, R. and R. Agrawal, "Mining Generalized Association Rules," *Proc. the 21th VLDB Conference*, pp. 407419, September 1995.
16. Srikant, R. and R. Agrawal, "Mining Quantitative Association Rules in Large Relational Tables," *Proc. ACM SIGMOD International Conference on Management of Data*, pp. 1-12, Montreal, Canada, June 1996.